Computer Graphics Forum (2013)
O. Deussen and H. Rushmeier (Editors)

Time Line Cell Tracking for the Approximation of Lagrangian Coherent Structures with Subgrid Accuracy

A. Kuhn[1], W. Engelke[3], C. Rössl[2], M. Hadwiger[3], and H. Theisel[2]

[1] Zuse-Institut Berlin, Germany
kuhn@zib.de
[2] Institute for Simulation and Graphics, University of Mageburg, Germany
wito.engelke@isg.cs.uni-magdeburg.de, roessl@isg.cs.uni-magdeburg.de, theisel@ovgu.de
[3] Geometric Modeling and Scientific Visualization Center, King Abdullah University of Science and Technology, KSA
markus.hadwiger@kaust.edu.sa

Abstract
Lagrangian Coherent Structures (LCS) have become a widespread and powerful method to describe dynamic motion patterns in time-dependent flow fields. The standard way to extract LCS is to compute height ridges in the Finite Time Lyapunov Exponent (FTLE) field. In this work, we present an alternative method to approximate Lagrangian features for 2D unsteady flow fields that achieves subgrid accuracy without additional particle sampling. We obtain this by a geometric reconstruction of the flow map using additional material constraints for the available samples. In comparison to the standard method, this allows for a more accurate global approximation of LCS on sparse grids and for long integration intervals. The proposed algorithm works directly on a set of given particle trajectories and without additional flow map derivatives. We demonstrate its application for a set of computational fluid dynamic examples, as well as trajectories acquired by Lagrangian methods, and discuss its benefits and limitations.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—time-dependent vector fields, flow field visualization, Lagrangian Coherent Structures (LCS), finite-time Lyapunov exponents (FTLE), time lines, material surfaces

1. Introduction

Lagrangian flow measures based on particle integration play an important role when analyzing time-dependent flow fields that are obtained from measurements or simulation. Those measures can be used to identify Lagrangian Coherent Structures (LCS) that efficiently encode features, such as flow transport barriers, and delineate regions of similar flow behavior [PD10]. One of the most prominent ways to extract LCS is via the finite-time Lyapunov exponents (FTLE) field as described by Haller [Hal01a, Hal01b]. LCS have been shown to be closely related to height ridges in the FTLE field, which is usually derived by approximating the local flow map gradient using finite-differencing schemes [Hal10]. The gradient of the flow map is known to contain strong variations, especially in areas where separation takes place. Hence, algorithms for its extraction rely heavily on a suitable density of sampled particle trajectories. A 'suitable density' usually has to be ensured by a dense grid for particle advection or by adaptive refinement strategies. In general, adequate values strongly depend on the intrinsic flow dynamics, such as amount of turbulence, as well as integration time. However, sufficient sampling rates are not always guaranteed (e.g., for very long integration intervals) or even possible (e.g., for Lagrangian acquisition methods, see Section 6). In this work we present a novel concept to approximate LCS via a geometric reconstruction of the flow map. Our work makes the following contributions:

- We introduce a method that works exclusively on a finite number of input trajectories. No additional sampling, field reconstruction, or refinement strategy is required.
- The produced features have subgrid accuracy, i.e., higher resolution than the original grid. They are guaranteed to be material structures w.r.t. the given discrete trajectory sampling.

To achieve this, our method maintains discrete, adaptive material lines within the advected grid structure. Those lines are used to reconstruct the flow map and estimate LCS more accurately on low resolutions compared to methods based on finite-differencing. We show that our approach works directly on different grid types and does not require an evaluation of additional flow map derivatives in Section 3. This makes it especially suited for (irregular) low spatial resolution grids and for applications where additional trajectory samples are either not possible, or not feasible. In addition, we provide a detailed outline on its efficient, memory friendly implementation (Section 4) and demonstrate its practical benefits and limitations (Section 6).

2. Related Work

To describe Lagrangian Coherent Structures (LCS), we typically observe properties of particles moving within the flow. This has been done for a broad variety of applications (see Peacock and Dabiri [PD10]). The extraction of LCS, based on the finite-time Lyapunov exponent (FTLE), has been popularized by Haller [Hal01a]. The computation of FTLE typically relies on an approximation of the spatial gradient $\nabla\phi$ of the *flow map* $\phi(\mathbf{x}, t_0, \tau)$. LCS are then derived as *height ridges* [Ebe96] in the resulting scalar field. We refer to Schindler et al. [SPFT11] for a comprehensive review of different ridge extraction techniques. Pobitzer et al. [PPF*10] further provide an overview of techniques for feature-oriented, time-dependent flow analysis. The classical FTLE method [Hal01a] approximates the flow map gradient in terms of finite differences. On regular uniform grids, this can be evaluated efficiently, while accuracy depends mainly on integration time τ and grid resolution. For this, a number of variations and extensions to this method have been presented: For fixed grids, better approximation quality may be achieved by using an auxiliary grid or additional samples. However, even on fine grid resolutions, neighboring particles in the vicinity of LCS can diverge rapidly, and reduce the quality of derivative estimates. Kasten et al. [KPH*09] propose a localized FTLE (L-FTLE) method that considers an infinitesimally small region by accumulating Jacobian matrices along one path line. Germer et al. [GOPT11] present an explicit and continuous reconstruction of binary level-sets that guarantee certain material properties. Üffinger et al. [USK*12] use higher order approximations for the flow map gradient. Their approach uses super-sampled elliptic circles (e.g., by using eight additional tracing samples per particle) to approximate the local deformation more accurately and to achieve better ridge quality.

FTLE Grid modification techniques. In order to obtain higher resolutions of Lagrangian structures while avoiding unnecessary samples, the initial grid structure has to be adapted or refined. One approach based on grid adaption is presented by Sadlo et al. [SP09]. They exploit spatio-temporal coherence by aligning the initial grid with the underlying LCS structures that are extracted from previous FTLE calculations. Sadlo and Peikert [SP07] present an

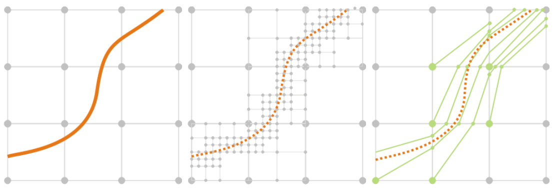


Figure 1: *Representation of material flow structures a) in different grids: b) an AMR uses additional samples [SP07], c) our method creates intersecting edges.*

adaptive mesh refinement (AMR) approach, based on separation rates obtained with coarse resolutions. It is stated that on sparse resolutions, LCS features can be missed, hence refinement is necessary in an offset area around identified features. While adaptive methods provide accurate computation of LCS for high resolutions [USK*12], they typically require additional samples on demand, as illustrated in Figure 1 b). To further reduce the number of samples, Agranovsky et al. [AGJ11] advocate the use of moving least squares (MLS) fitting. A variant for computations in adaptive, irregular manifolds is proposed by Lekien et al. [LR10].

FTLE acceleration techniques. The high spatio-temporal coherence of LCS structures can be used to speed up the computation of FTLE sequences. Garth et al. [GGTH07] and Brunton and Rowley [BR10] focus on avoiding redundant computations in this case. Lipinski and Mohseni [LM10] present an algorithm to extract and track FTLE ridges in sequential FTLE computations. Their method can be used to predict the behavior of LCS structures over time, if previous FTLE fields are available in adequate resolution. Hlawatsch et al. [HSW11] present a method to directly interpolate new hierarchical path line segments, based on a finite set of available path line segments. Trajectory interpolation can be used in cases of sparse sampling rates, but can also lead to displacement or distortion of Lagrangian quantities which may be hard to predict a priori [SFBP09].

Alternative LCS extraction approaches. More recent work by Haller [Hal10] has shown that LCS and FTLE ridges are not necessarily correlated. Based on this, Farazmand and Haller [FH12] present the concept of *strain lines* to directly describe LCS without evaluating FTLE. This concept has the advantage of obtaining well parametrized LCS curves with a significantly reduced number of samples. Still, initial grid resolution and flow map derivative approximation require thorough consideration to avoid missing or distorting features. Besides, flow map gradients are especially sensitive close to LCS structures, as shown by Olcay et al. [OPK10] and Kuhn et al. [KRWT12]. Another alternative is presented by Sadlo and Weiskopf [SW10], who showed how to derive topological features for 2D unsteady flows by replacing the role of stream lines by streak lines. This concept has been extended to 3D unsteady attracting and repelling manifolds by Üffinger et al. [USE12].

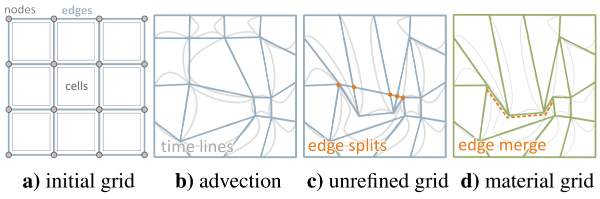


Figure 2: *Illustration of material edges in a discrete grid to approximate the behavior of continuous time lines. The refinement of the initial grid introduces* ***split*** *and* ***merge*** *events.*

All previous methods take a flow as input, and may thus not apply to applications, where there is no flow field available. For instance, Thiffeault [Thi10] analyze oceanographic flow data based on a finite number of measured trajectory data. It is stated as a fact that especially for measured particle-based phenomena hardly any dense flow field data is available. Furthermore, sparse optical flow methods, such as presented by Senst et al. [SES10, SEHS11], avoid the expensive computation of a dense flow field, and only store a finite set of trajectories around relevant features.

3. Material Line Advection in 2D Flow Fields

The flow map $\phi(\mathbf{x}, t_0, \tau)$ describes the time-dependent motion and reorganization of particles in the flow over time. Typically, the flow map is represented as a regular grid that is advected with the flow field. The approximation of derivatives by finite-differencing assumes straight lines as connections between neighboring particles in the initial grid. More advanced concepts, introduce additional samples to improve derivative approximation (e.g., advection grids [SRP11], AMR [SP07]) or perform local reseeding if separation exceeds a certain threshold (e.g., localized FTLE [KPH*09]). However, all of the above mentioned methods assume straight lines as connections, which is only a rough approximation of the actual flow behavior as shown by Üffinger et al. [USK*12] (examples see Figure 3 b) and Figure 8). In fact, in a continuous view, a line between two particle trajectories behaves like a *time line* during advection. Time lines require frequent refinement to be adequately resolved (similar to streak lines [USE12, WT10, WWSS10]) and act as material boundaries over time, i.e., no particles can cross them. The core aspect of our approach is to maintain this *material condition* of initial edges between neighboring particles based on the available trajectories. The resulting grid approximates the geometry of time lines, and hence the flow map, more accurately than previous schemes. Further, split and merge events (Figure 2) give more detailed structural information about material lines *without* additional samples.

3.1. Time Lines and Lagrangian Features

In literature the relation of streak lines (and streak surfaces) to LCS has been discussed in detail by Sadlo et al. [SW10] and Üffinger et al. [USE12]. Similarly, time lines are suitable indicators to Lagrangian features in unsteady flow fields

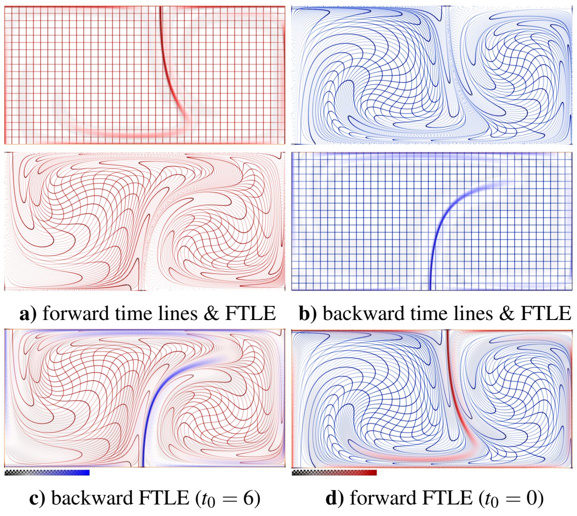


Figure 4: *Time lines and FTLE in the double gyre flow [Sha05]: Regular time line grid advected in a) forward and b) backward direction for $\tau = 6$; c) shows the forward grid together with the backward FTLE ($t_0 = 6$); d) backward grid and forward FTLE ($t_0 = 0$) with emphasized ridges.*

as shown by Weinkauf and Theisel [WT12]. More specifically, they can be shown to be closely related to LCS structures in the flow field: Time lines are material structures by definition, while LCS are special time lines maximizing Lagrangian properties, such as separation [Hal01a], repulsion [FH12], or hyperbolicity [SW10]. In areas of low separation, time lines roughly maintain their sampling density and orthogonality with respect to their original neighborhood. In contrast, time lines crossing forward LCS and integrated in forward direction are subject to strong deformation (due to separation and/or shear). At the same time, Sadlo and Weiskopf [SW10] have shown that time lines seeded exactly on a saddle-type LCS *collapse* to a single point. For the same reason, backward time lines have to align with dominant forward LCS structures in this case: If the (collapsed) saddle line is enclosed by time lines at time step t_0, the same time lines have to enclose this structure at any other time step $t_0 - \tau$, since both structures are guaranteed to be material structures, i.e., cannot cross each other. This effect is illustrated in Figure 4 for the double gyre example [SLM05]. Hence, one option to identify such LCS, is to identify time lines that collapse to a single point under forward advection. However, this is a non-trivial task, since it requires a dense time line seeding across the complete domain, while sufficiently dense sampled time lines cannot intersect at all.

3.2. Flow Map Approximation using Time Lines

To still be able to find such 'collapsing' time lines, one has to define a *discrete* representation of the flow map, in which such events can occur, and hence allow for an approximate view on LCS. For this, we explicitly define the initial connections between particles as neighborhoods in a sampling

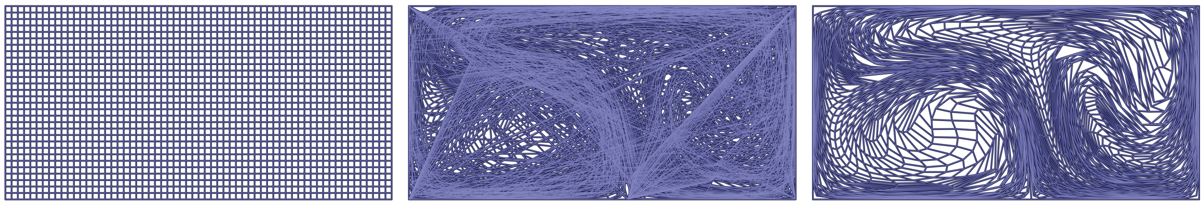

a) initial grid configuration
b) unprocessed grid
c) grid with material condition

Figure 3: ***Double gyre:*** *Figure a) shows the initial quad grid with* 64×32 *cells at* $\tau = 0$ *in, b) the unprocessed advected grid at* $\tau = 10$*, and c) with tracked material condition at* $\tau = 10$.

grid: we assume that the domain is partitioned into a cell complex $G = \{N, E, C\}$, which consists *nodes* N, *edges* E, and *cells* C (see Figure 2). *Nodes* refer to single particles in space-time domain that are advected within the flow, capturing the geometric behavior of the flow field. The trajectory of each node forms a path line. *Edges* connect nodes within the initial grid and express spatial relations between nodes. In a continuous view, their advected instances act as time lines, which require local refinement with ongoing advection (i.e., see Figure 4 a) and b)). Also in this setting, every time line is a material line over the complete time interval, hence nodes cannot cross it. *Cells* are bounded by a set of adjacent edges. The complexity of their shape increases significantly with ongoing integration and usually requires additional refinement (especially in the vicinity of LCS). In incompressible continuous flows, cells further don't change their initial volume, but undergo a potentially significant stretching and shearing [FH12].

3.3. Material Constraints for Discrete Time Lines

In order to maintain the discrete time line material constraint using this cell complex, we have to identify events where this constraint is violated, i.e., identify particles that cross edges. This corresponds to *intersections* of linear edge segments with particles moving within the flow (see Figure 2). By preserving this material constraint, based on the available samples during integration, one can derive additional geometric information w.r.t. to the initial grid:

- Nodes intersecting edges generate two types of events:
  First, a *split* event creates two new edges. This increases the geometric complexity of the initial edge segment: an edge becomes a polyline
  Second, a *merge* event collapses two existing edges that share the same nodes into one edge.
- The sequential order (w.r.t. to τ) of split and merge events determines the temporal evolution of the grid edges, i.e., discrete time lines. Although an error is induced by the assumption of linear edge segments, the reconstructed cells capture the real, continuous behavior more accurately than unrefined grids (see Section 3.5).
- Edge crossings generate additional connectivity information within the initial grid cells: If an intersection has been determined at a time step τ_i, we can easily identify the corresponding initial edge in the grid configuration at t_0. Sorting all intersection events for one node, and connecting them in the initial grid at t_0, results in a set of additional edges: We denote those edges as **virtual edges**. They are directly associated with an intersecting node and an advection time τ_i. The set of all virtual edges for one node forms a polyline crossing cells and is denoted as **virtual path**. By construction, the polyline formed by a virtual path is guaranteed to collapse to a single node after the considered integration interval. More details are given in Section 3.6.

In order to detect split and merge events and reconstruct the resulting topological information, we split the computation into an *advection* stage and a *reconstruction* stage.

3.4. Discrete Grid Advection

In the advection stage, all nodes are integrated in the given 2D time-dependent vector field to generate particle trajectories, i.e., path lines. Let $(\mathbf{x}, t) \in \mathbb{R}^2 \times \mathbb{R}$ be a domain point in space-time, and let $\mathbf{v}(\mathbf{x}, t)$ denote the time-dependent flow field. Then a path line $\mathbf{p}(\mathbf{x}, t, \tau)$ is defined as the solution to

$$\frac{\partial}{\partial \tau}\mathbf{p}(\mathbf{x},t,\tau) \;=\; \mathbf{v}(\mathbf{p}(\mathbf{x},t,\tau), t+\tau) \;\text{ with } \mathbf{p}(\mathbf{x},t,0) = (\mathbf{x},t) \;.$$

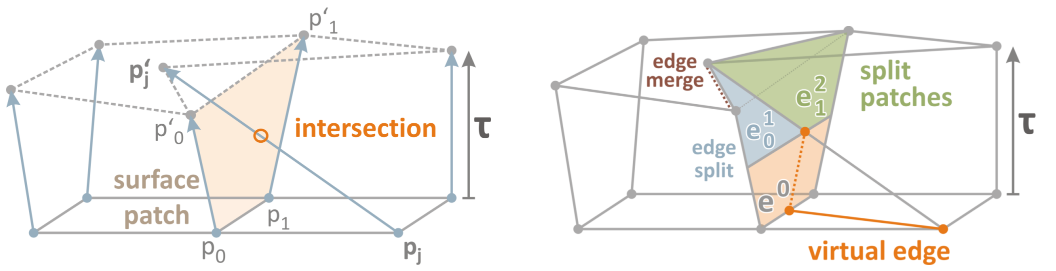
We assume *discrete* path lines that are represented as piecewise linear curves with point samples $\mathbf{p}(\mathbf{x}, t, k\Delta\tau)$ for non-negative integers k and a sampling interval $\Delta\tau$. Let $\mathbf{x}_i$ denote the position of node $i \in N$ in the initial grid at integration time $\tau = 0$. We write $\mathbf{p}_i(\tau) := \mathbf{p}(\mathbf{x}_i, t, \tau)$. Then the discrete path lines consist of linear segments

$$\mathbf{p}_{ik}(\xi) \;=\; (1-\xi)\,\mathbf{p}_i\big(k\Delta\tau\big) + \xi\,\mathbf{p}_i\big((k+1)\Delta\tau\big)\;,\;\; \xi \in [0,1]\;.$$

In practice, the continuous path line $\mathbf{p}(\mathbf{x}, t)$ is computed as a piecewise polynomial curve, whose order and smoothness depends on the numerical integration scheme. The discrete path line with linear segments $\mathbf{p}_{ij}$ is obtained by a uniform resampling of this curve w.r.t. τ. Figure 3 shows an example for an advected rectangular grid in the double gyre [SLM05] for $\tau = 10$, with and without the given material condition.

3.5. Edge Intersection Reconstruction

In order to detect violations of the material constraint, we have to check edges of the advected grid for intersections: Let E_k be the set of edges at integration time $\tau_k = k\Delta\tau$ with $E_0 = E$. We iterate over discrete time steps $\Delta\tau$. At every time,



a) grid advection process **b)** edge reconstruction

Figure 5: *Conceptual representation of the splitting procedure: Moving edges form bilinear surface patches that are tested for intersections with moving particles (i.e., lines or rays). If an intersection occurs, the original edge is split into two edges and surface patches and creates a* ***virtual edge****.*

we can process each edge $(i,j) \in E_k$ individually, which is spanned by the linear segment

$$\mathbf{e}_{ijk}(\eta) \;=\; (1-\eta)\,\mathbf{p}_i(\tau_k) + \eta\,\mathbf{p}_j(\tau_k)\,, \quad \eta \in [0,1]\,.$$

Here, indices $i,j \in N$ denote nodes, k refers to a time step. Then moving the edge (i,j) within the interval $[\tau_k, \tau_{k+1}]$, along its spanning trajectories results in a *bilinear surface patch*

$$\begin{aligned}\mathbf{s}(\eta,\xi) \;&=\; (1-\xi)\,\mathbf{e}_{ijk}(\eta) + \xi\,\mathbf{e}_{ij(k+1)}(\eta)\\ &=\; (1-\eta)\,\mathbf{p}_{ik}(\xi) + \eta\,\mathbf{p}_{j(k+1)}(\xi)\,.\end{aligned}$$

Here, $\eta \in [0,1]$ is the parameter position on the edge (i,j) and refers to the spatial domain, and $\xi \in [0,1]$ provides a reparametrization of the time interval $[\tau_k, \tau_{k+1}]$. Checking for a node $\ell \in N$ crossing the edge (i,j) refers to testing for an intersection of $\mathbf{s}$ with the discrete path line segment $\mathbf{p}_{\ell k}(\nu), \nu \in [0,1]$. For the 2D case, this operation has been studied intensively for ray tracing, and there are efficient numerical algorithms for computing this intersection [RPH04]. At each time step, we apply intersection tests for every edge and every node before advancing to the next time step. Edge intersections create split events (and possibly merge events if duplicate edges occur), which lead to a local refinement of the set of edges E_{k+1} and hence of the cell complex (see Figure 5). This refinement is based purely on the given trajectories and our material condition, its is thus different from standard adaptive mesh refinement strategies (see Figure 1).

3.6. Cell Reconstruction

The initial (intersection-free) grid cells C are bounded by edges in E. Every detected intersection, indicates that a particle is entering or leaving a cell at a certain time. Connecting both events, entering and leaving, forms a new edge (virtual edge) across the cell and divides the cell into two parts. Due to the fact, that virtual paths approximate collapsing time lines, we assume that they are suitable estimates of the position and shape of crossing LCS w.r.t. to the available trajectories. However, due to the finite number of samples and the discrete nature of our scheme, only a subset of those structures is relevant. The most dominant virtual paths can be identified by their length (i.e., sum of corresponding virtual edge lengths at t_0), since those have been subject to the locally strongest deformation. This is similar to the filtering of FLTE ridges, e.g., by separation strength [SPFT11]. Note that the particular topology of G at $\tau = 0$ can be chosen arbitrarily, e.g., a regular grid or a Delaunay triangulation. In the following, we use convex quads and triangles similar to existing methods (such as Lekien et al. [LR10]).

4. Implementation

The algorithm to conduct the described procedure can be summarized as follows: The advection part takes a set of trajectories $\mathbf{p}_i$ and an initial cell complex G as input. In case we are given a flow field and seed points, particle advection is performed in a preprocess. In our implementation we use an adaptive fourth-order Runge-Kutta integration. Similarly, if only the start points of the trajectories are given and there is no cell complex G defined, we generate G as the Delaunay triangulation of the given nodes. Depending on the application, it may become advantageous to artificially bound the domain (or fill parts of it) by inserting additional points at zero flow regions to recover the interaction of the boundary with domain boundaries or obstacles (e.g., see cylinder example in Figure 8). The reconstruction part proceeds in two stages: First, it determines edge splits and merges (Sec. 3.5):

```
$E_0 := E$
**for all** time steps $k \geq 0$ and $k\Delta\tau < \tau$
  $E_{k+1} := E_k$
  **for all** edges $(i,j) \in E_k$
    **for all** nodes $\ell \in N \setminus \{i,j\}$
      **on** patch/path line segment intersection
        record event
        split/merge edge and update $E_{k+1}$
```

Second, all detected split events are inserted to the initial grid, by adding the resulting virtual edges and virtual paths (Sec. 3.6). This is done by sorting all events by ascending time stamps τ_i and then finding corresponding pairs of events that indicate when a cell was entered and left (see Section 3.6). The nodes associated with such pairs are connected and form a virtual edge that splits a cell. Figure 5 illustrates the intersection and splitting step, and Figure 8 shows a result of the algorithm for the cylinder example. We use the algorithm in [RPH04] to perform intersections between a line segment and a bilinear surface patch. In case of multiple intersections, we sort all events after τ, process the first event, and check for intersections with the two new surface patches (see Figure 5). In order to speed up computations, a bounding box test is applied first, and only line segments passing these tests are considered for intersection testing.

Performance. For all our experiments we used single core of an Intel i5 CPU with 2.7GHZ and 8GB of RAM. The required computation time for the double gyre example is shown in Figure 10, and for the cylinder flow in Figure 17. Without further optimization (e.g., via spatial hashing), we have to test every edge with every edge in the material grid for intersection. In this case, the required number of tests grows exponentially with the number of trajectories and advected grid edges (Fig. 17 c)). The memory requirements are directly derived from the number of trajectories with the temporal sampling rate $\Delta\tau$ (Section 5), number of edges in the advected material grid, and the overall number of intersection that occur during advection. The number of trajectories and $\Delta\tau$ are fixed and are given with the application or determined automatically. Our experiments indicate that number of front edges remains roughly constant, or even decreases over time, Fig. 10), due to frequent merge events. The main factor is the number of intersections that occur, which depends on the flow characteristics. For our experiments the amount of intersections typically shows a linear behavior with increasing integration time. Note that we can either store all intersections and reconstruct the virtual edges in a post-process, or omit intersections and reconstruct virtual edges on the fly.

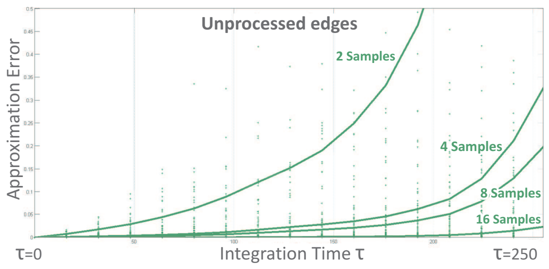
5. Verification and Discussion

In the following, we describe experiments for the verification of our method, and we evaluate accuracy and convergence as well as grid dependence. Our method has the following parameters: the integration time τ, the temporal sampling rate $\Delta\tau$, and the initial grid of nodes (which determines connectivity and the spatial sampling rate). We test its behavior w.r.t. τ and the grid.

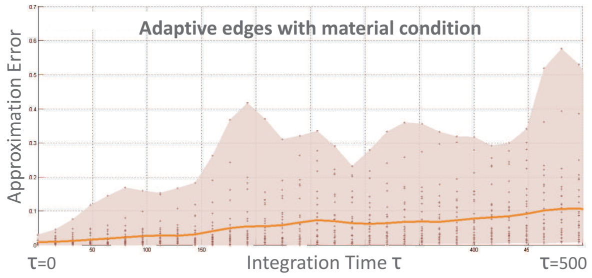
Sampling interval $\Delta\tau$. The parameter τ and the initial grid is provided externally by the user, and is often inherent to the data or application. In contrast, the temporal sampling interval $\Delta\tau$ for uniform resampling of trajectories is a parameter to the algorithm (see Section 3.5). Increasing the $\Delta\tau$ decreases the number of required iterations in the outer loop and improves performance, however, at the risk of missing intersection events. In contrast, very small values, i.e., becoming "more time-continuous", do not necessarily improve the results due to the coarse spatial resolution, but increase the number of required intersection tests. Our experiments suggest that $\Delta\tau$ can be set conservatively to balance performance and accuracy: For our examples we use $\frac{1}{10}\times$ the ratio of average edge length and maximum vector magnitude, and observe no significant change in the results by a further reduction.

Integration Time τ. As any other Lagrangian approach our method depends on the length of the (integrated) sample trajectories. In our case this length is directly controlled by the parameter τ. Commonly, any errors amplify during integration, which can lead to significant deviations after longer integration intervals [BJB*11, KRWT12]. We compute discrete time lines during integration, hence the deviation from the "real" continuous time lines affect the resulting LCS approximation. To test the dependence on spatial resolution we define the following experiment: given is a flow field, and we consider a single linear edge from a given initial grid. Every edge forms an approximation of a discrete time line and we assume to get a better approximation by using additional samples. We remark that this may not be an option in practice, since additional measurements may be required [Thi10, SFBP09]. The original edge uses 2 samples, and we increase this number exponentially using $4, 8, 16, \dots$ samples of the initial straight line segment at $\tau = 0$. Then we observe the resulting time lines at integration times $k\Delta\tau$. Finally, we compare these time line approximations of different accuracy to a highly refined version (2000 samples at $\tau = 0$) by computing the sum of the minimal distances between both polylines (i.e., we used pairwise root mean square (RMS) error distances of each node in both polyline sets). Figure 6 shows the averaged results for the double gyre data set for 30 random edges on a resolution of 20×40. At a fixed sampling rate, the error grows exponentially after a certain integration time (Figure 6, left). In contrast, edges that are adaptively refined using our material constraint remain more accurate for longer integration times. Note that edges crossing LCS tend to produce highest error rates.

Grid Dependence. Since our methods solely depends on discrete lines as cell boundary, it is not restricted to a particular type of grid structure (i.e., regular or irregular). We analyze the dependence of results on the grid by applying a series of rigid transformations on a sparse, regular, and rectangular grid with cell size h. First, the grid is translated in both directions by a offset in $[0,h]$, it is then rotated by an angle in $[0, \frac{\pi}{4}]$. After the transformation we extracted all LCS approximations, i.e., virtual paths, and highlighted the 10% longest features. Figure 7 shows the result for a series of 30 random transformations and the resulting approximation for a 20×40 grid on the double gyre example. The observable variations mainly depend on local cell size and especially occur in regions where the grid edges and LCS direction are nearly parallel, i.e., intersections occur at very small angles. Overall, the extracted virtual paths still give a suitable approximation of the underlying LCS structure. This holds especially when considering FTLE based on finite differences for sparse resolutions (Figure 11 a)).



a) time line distance for $2,4,8,16$ samples



b) time line distance of edges with material condition

Figure 6: ***Double Gyre***. *Figure a) shows the average deviation of* 30 *random edges with* $2,4,8,16$ *samples per edge from the same time lines sampled densely with* 2000 *particles, Figure b) same with refinement by our material condition. The colored region denotes the min-max range of the error among all samples.*

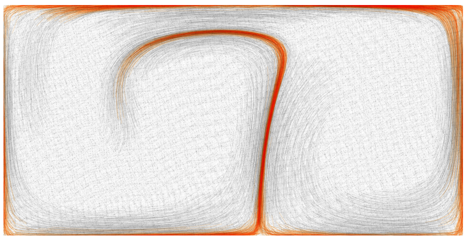
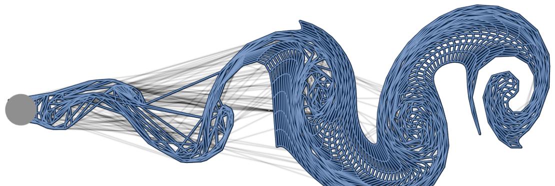
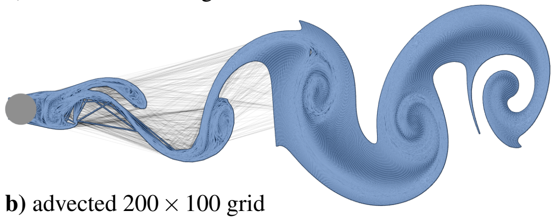


Figure 7: ***Double Gyre***: *Stacked LCS approximations for* 30 *grids that have been displaced by random rotations and translations. (*20×40 *grids and* $\tau = 10$*). The* 10% *longest features have been emphasized (orange).*



a) advected 50×25 grid



b) advected 200×100 grid

Figure 8: ***Cylinder Flow***: *Flow map grids with material condition (blue) and unprocessed edges (grey) at* $\tau = 120$.

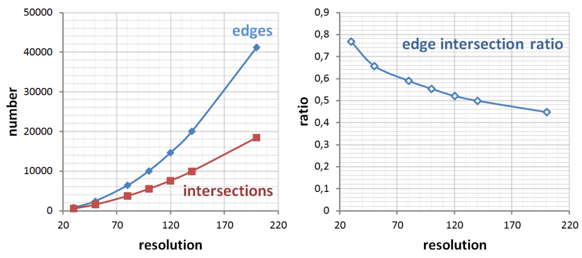


Figure 9: ***Cylinder Flow:*** *Plot of the number of edges with increasing resolution (left) and average intersections per edge (right, c.f. Figure 17).*

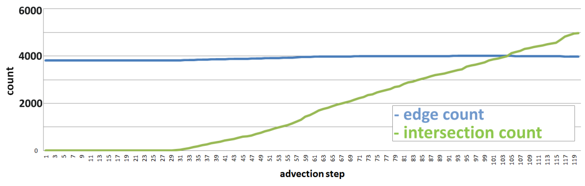
Resolution Dependence. To show the impact of the resolution to our method we run our algorithm on the cylinder data set with varying resolution (see Figure 16). In this case, our method achieves adequate results with up to factor 10 fewer samples, than FTLE on similar resolutions. The risk of missing important features on low resolutions is significantly reduced (Figure 16, first row). This is especially relevant in comparison to AMR methods [SP07] that require to approximate LCS locations on low resolutions for further refinement. Sparse particle approaches, based on MLS fitting [AGJ11], are able to achieve excellent results on reduction factors up to 4. However, they have been shown (visually) to potentially displace or distort features on equally low sampling rates, and may require adaptive regular resampling of particles. Figure 17 shows the resulting cell complex elements with timings and Figure 8 two examples of advected material grids at different resolutions. It is especially interesting to observe that the number of required material grid edges (front edges, Fig. 17 a)) only increases slowly, and the number of occurred intersections grows at linear rates (compare Fig. 10 a) and Fig. 17 b)). In comparison: the straightforward refinement of time lines (Fig. 6 a)) may require an exponential amount of additional samples, especially in areas of strong deformations. This can behavior is confirmed by the number of average intersections per edge with increasing resolution in Figure 9 and 17 b). The edge intersection rate reduces with increasing amount of samples. The indicated trend leads to the assumption that an average intersection rate per edge converges against a specific value, but does not vanish completely. For comparison: the 200×100 grid in Fig. 16 already uses 20 times more samples than the original flow field in the same region (about 40×20). However, there are cases (see Section 6) where such sampling rates are not feasible or possible.

Limitations. We remark that the quality of the initial grid, i.e., the shape of cells, has significant impact on our method. The patch-ray intersection [RPH04] is generally very stable. However, degenerated cells or intersections at small angles can introduce inaccuracies that result in discontinuities along virtual paths (see Fig. 16, top row). Another important aspect is, that in general, we do not assume the LCS approximations to re-simplify during the flow integration process. On the same note, our approach does not cope with moving or vanishing structures with increasing τ. In general, to accurately resolve such cases requires additional trajectory samples. The same applies to vector fields containing divergence, that produce spatial undersampling over time. Still, the geometry of advected cells and number of split events in the graph structure is a suitable indicator to detect such areas, e.g., for further sample refinement. Finally, the structures identified with our scheme are based on the assumption that LCS can be approximated by discrete collapsing time lines based on the available trajectories. Although we show empirically that this applies to a variety of scenarios, there is no formal proof of this property at that time. There are examples where our scheme can give larger number of potential false-positives (e.g., due to noise, see Figure 15) or deviates from alternative LCS definitions (e.g., for rotational shear flows [Hal10]).

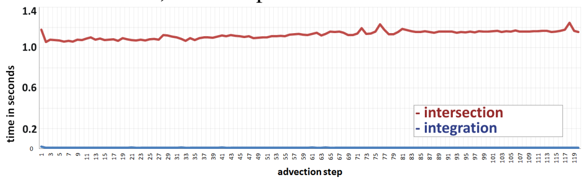
6. Results

We demonstrate the effectiveness of our method for a number of flow fields that are either synthetic benchmarks or stem from numerical simulations. All FTLE fields that are shown for comparison were computed with Haller's [Hal01a] classic algorithm. The first three examples use an underlying smooth velocity field from which a low number of path lines are sampled. Clearly, for such data sets better LCS structures can be obtained simply by sampling more path lines (e.g. by using AMR [SP07]). We use this as a ground truth to compare our LCS approximations. However, the last two examples (optical flow and SPH) do not come with an underlying field so our method is the only one available that achieves subgrid accuracy.

Double Gyre Data. The first example is the double gyre example introduced by Shadden et al. [SLM05]. This synthetic data set is frequently used to illustrate the performance and suitability of FTLE approaches such as strain lines [FH12] or material separation fields [GOPT11]. We consider integration times $\tau = 12$ and $\tau = 20$. Figure 11 shows FTLE computed on a low resolution 64×32 grid (Fig.11 a,b) compared to our method on the same resolution. The last row shows an overlay with FTLE computed on a 512×256 grid. The path id is color coded so that different paths can be distinguished. For this example, our approach captures the most prominent ridge structures, until $\tau = 20$ even for a very coarse sampling. In contrast, FTLE on a comparable number of trajectories fails to capture major LCS structures as can be seen in Figure 11 d). The double gyre data is also shown in Figure 3 to illustrate the effect of our material condition, and in Figure 7, 10, and 6 for experiments on accuracy, performance and robustness (see previous Section 5).



a) cell complex element count



b) timing

Figure 10: ***Double Gyre****. Figure a) shows the number of edges in the material grid and number of intersection events, b) shows the time required for intersection and integration for each time step for the* 64*x*32 *grid shown in Figure 11.*

Cylinder Flow. The cylinder flow field represents a simulated fluid flow that has been created using the Free Software *Gerris Flow Solver* [Pop04]. Its temporal evolution is dominated by a periodic vortex shedding which leads to the famous von Kármán vortex street. The alternating recirculation structures cause strong flow stretching especially behind the cylinder. Figure 12 shows the FTLE fields and results of our method: The upper half of the topmost image shows FTLE computed on a high resolution 300×700 grid. The lower half compares to a 40×180 grid, which is less than 4% of the high resolution samples. On low resolutions, features are blurred and are likely to be distorted or missed in a ridge extraction. In contrast, our method produces sharp and detailed features even for low resolutions as shown in the center. The close-up at the bottom reveals that shape, number, and location of the LCS are clearly recognizable. Colors of virtual edges encodes the length of the associated virtual path, i.e., length of the corresponding polyline at t_0 (see Sections 3.3. Note that, we use a regular triangulation of the underlying grid. The advected and corrected grid at two different resolutions is shown in Figure 8.

Heated Cylinder. The second simulated data set captures the natural convection generated by a heated cylinder using the Boussinesq-approximation to generate the turbulent vortex behavior (simulated using *Gerris Flow Solver* [Pop04]). In the space-time domain it starts with a stagnant fluid, while due to heating a highly turbulent plume develops quickly above the cylinder, which becomes apparent in Figure 14.

Figure 14 a) and b) shows an FTLE field and compares to a virtual edge triangulation computed by our method. The sequence has been computed for a fixed $\tau = 20$ and different $t_0 = 20, 30, 40$. Our algorithm can use significantly less samples as input, while virtual paths still reveal distinct and well resolved LCS features. Although the FTLE field uses $\approx 2.7\times$ the number of samples, features get lost and can only be reconstructed using higher sampling rates (e.g., by

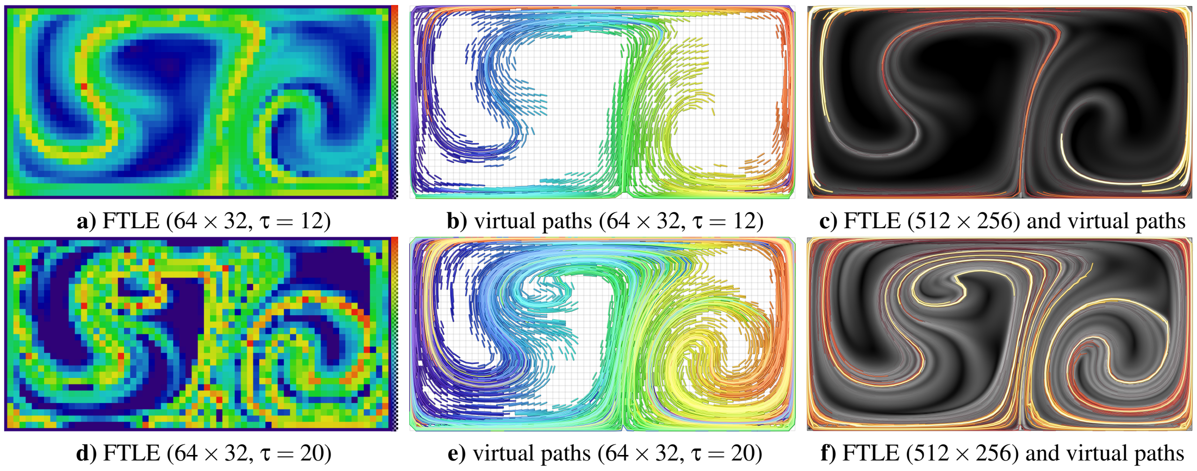


Figure 11: ***Double Gyre:*** *a) and d) FTLE with* 2048 *samples ; b) and e) our results with material condition displaying all virtual paths in the base grid (colored by node id) ; c) and f) comparison of FTLE with* 131072 *samples and virtual paths using* 2048 *samples (colored and scaled by length).*

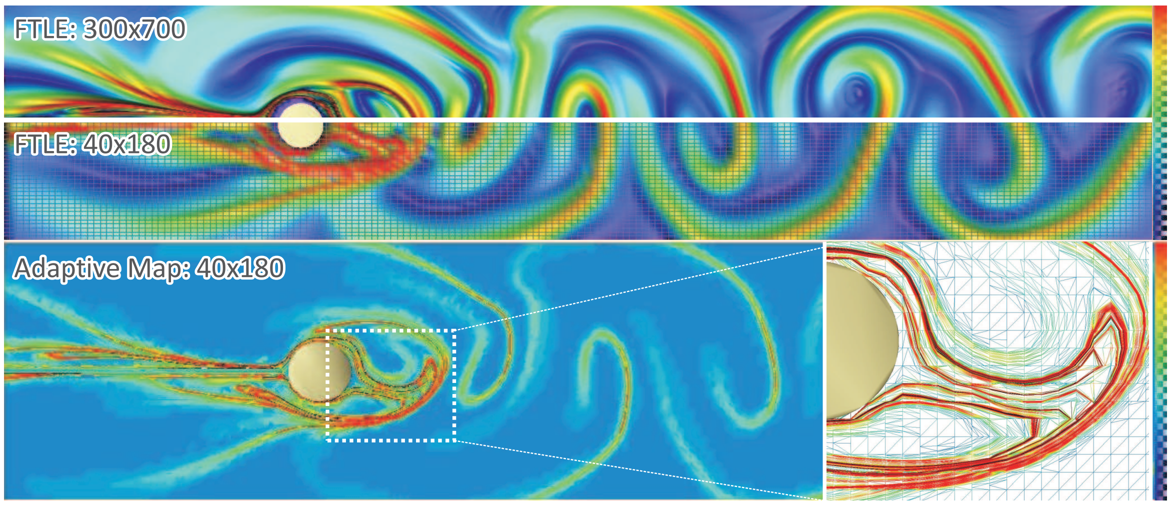


Figure 12: ***Cylinder Flow*** **(**$\tau = 120, t = 0$**).** *Top: FTLE for original flow field resolution* (300×700) *and low resolution* (40×180)*. Center: our result ; Bottom right: Adaptive map, i.e., retriangulated virtual edges in the initial grid, with* 40×180 *samples, colored by virtual path length. Bottom left: close-up of our resulting adapted mesh colored by edge length.*

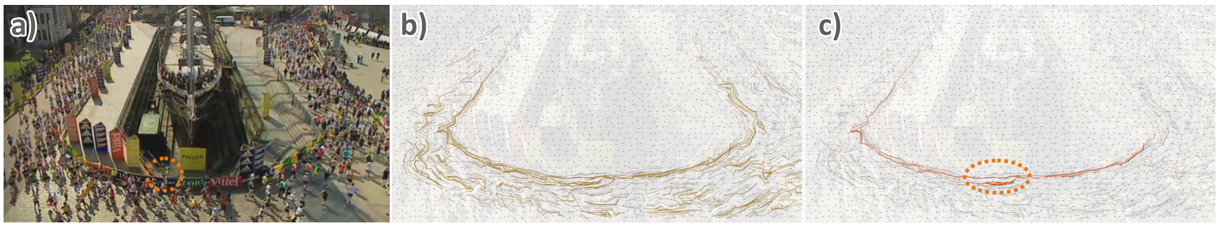


Figure 13: ***Marathon Sequence:*** *Our method applied to a sparse irregular sampling computed from a marathon video sequence. Image a) shows the first video frame, b) the 80% longest features, and c) the 10% longest features extracted. A group of persons blocking running area creates a separation irregularity along the cordon which has been highlighted in a) and c)*

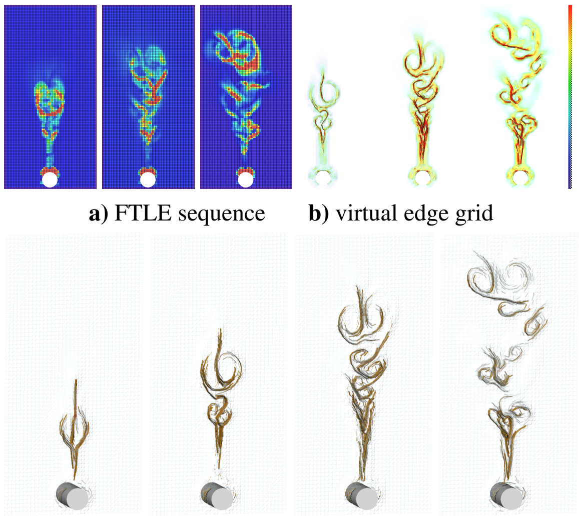


Figure 14: ***Heated Cylinder:*** *Time sequence comparison for FTLE* (60×120) *and triangulated virtual edge grid with* 36×72 *and fixed* $\tau = 20$ *and* $t_0 = 20, 30, 40$.

using adaptive refinement [SP07] or higher-order approximation [USK*12]). Features extracted with our method are represented by distinct polylines, illustrated in Figure 14 c), and have been scaled and colored by their length.

Optical Flow Field: Marathon Sequence. Until now, we extracted LCS from given flow fields. One particular feature of our method is that LCS can be extracted right away from given trajectories without need to evaluate the flow. This opens applications to scenarios, where only trajectories are measured. One example is optical flow data, and we apply our method to the *marathon sequence* benchmark data. To segment crowd video data, FTLE has been successfully used for dense optical flow data by Ali et al. [AS07]. However, computing dense optical flow fields is computationally expensive and does not necessarily exploit the adaptive nature of the underlying motion. To speed up computation and to avoid oversampling *sparse methods* have been proposed, e.g., by Senst et al. [SES10, SEHS11]. Sparse methods compute a set of irregularly distributed trajectory samples to represent motion in the underlying image sequence. In this example, we apply our method to a set of sparse trajectory data of a marathon sequence. The original video data has a resolution of 720×404 pixels, while the optical flow field trajectories have been computed for 297 frames. For our evaluation we used a 2040 trajectory samples, with an initial random distribution. The resulting feature approximations obtained with our method are shown in Figure 13: a) shows the first frame of the video sequence, while b) and c) show the 80% and 10% longest virtual path structures, respectively. The features clearly separate areas of strong motion separation and characteristic patterns within the motion field. In this case, for example, a group of persons is standing close to the

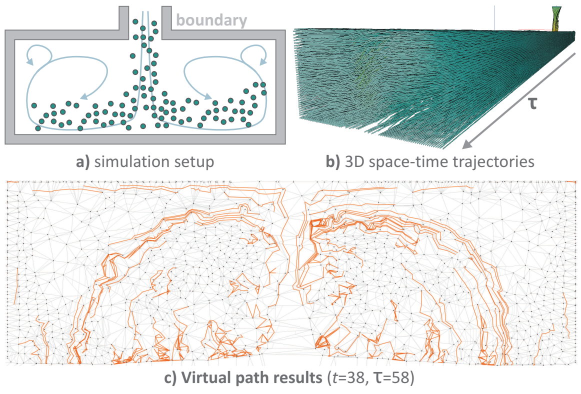

boundary
τ
a) simulation setup
b) 3D space-time trajectories
c) Virtual path results (t=38, τ=58)

Figure 15: ***SPH Simulation:*** *Slice of the SPH simulation at* $t = 38$. *a) shows the simulation setup and the domain, b) a rendering of the trajectories in space time, and c) the base grid with virtual path structures (light grey, 10% longest highlighted in orange,* $\tau = 58$*)*

cordon of the running area, which creates an irregularity in the virtual paths due to people running around the blocked area (highlighted in Figure 13).

SPH Simulation Data. Two examples for Lagrangian acquisition methods are Smoothed Particle Hydrodynamics (SPH) data for simulation (e.g., [SFBP09, JFSP10]), and optical flow measurements from Particle Tracking Velocimetry (PTV) data (e.g., [RGPS05]), which both produce a finite number of particle trajectories only. Although further interpolation of trajectories is possible, it is non-trivial to decide which interpolation methodology avoids displacement or distortion of the features in the underlying data [SFBP09]. In fact, [SFBP09] show that interpolation between trajectories of an SPH data set may not give satisfactory results for the detection of LCS. Hence, we applied our method using 2D Lagrangian simulation that has been done using a state-of-the-art SPH tool and contains about 5000 particles filling up a box (Figure 15 a). After the simulation the geometry of every particle trajectory was exported (see Figure 15 b)). We sliced all available trajectories at $t = 38$, created an initial grid using Delaunay triangulation, and reconstructed the cell complex for $\tau = 58$. The results of our approximation are shown in Figure 15 c) (virtual paths and initial grid in gray, 10% longest paths are highlighted). The virtual paths are less smooth than in the first three examples, since trajectories of SPH simulations are not necessarily as smooth as integral curves of a dense velocity field. However, the approximation still reveals the basic Lagrangian behavior and indicates two dominant symmetric LCS structures around the inlet, and the two major vortex structures. The approximations provide a basic feature-oriented analysis of trajectory data, that would have been inaccessible to previous methods.

7. Conclusions

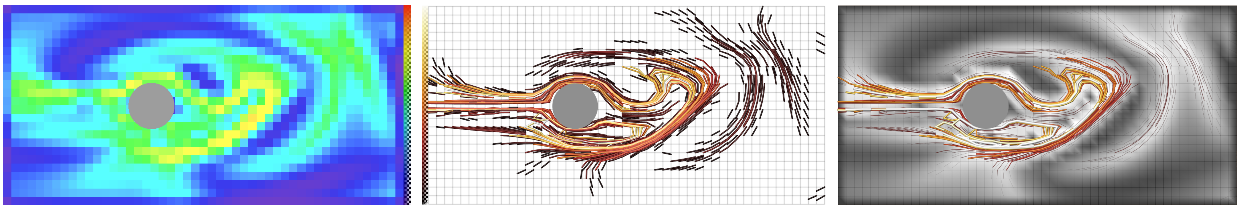
In summary, we present a novel approach to estimate LCS using a discrete material constraint during advection. We introduced a discrete cell graph structure to approximate the flow map geometry and to reconstruct material lines w.r.t. to the available samples. The presented concept provides the following benefits:

- LCS approximations are a direct result of our algorithm and are obtained as piecewise linear curves (virtual paths). In contrast to existing FTLE methods they require no post-processing such as ridge extraction [Ebe96, SPFT11].
- No derivatives of the flow field or its flow map are required explicitly. In fact, the result solely depends on the given trajectories and our material constraint.
- The method requires a significantly lower number of samples than comparable approaches. It has been demonstrated that it achieves adequate LCS estimations in subgrid accuracy, even for sparse and irregular grids
- The method allows for efficient reconstruction of discrete time lines and reduces refinement to a graph traversal problem.

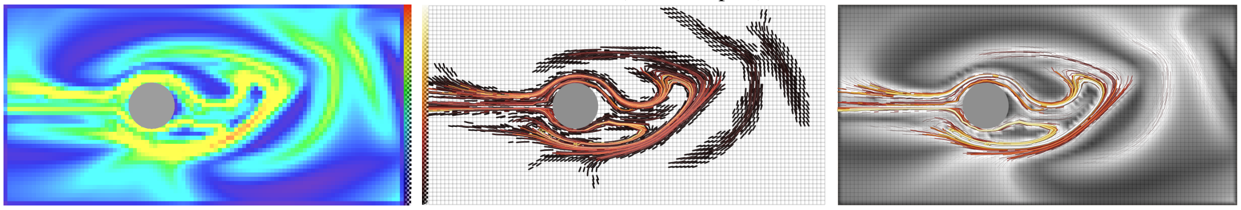
In conclusion, the presented algorithm is especially interesting in applications, where there are only a finite amount of measured or simulated particle trajectories, an irregular base grid, or no dense continuous description of the underlying vector field available (e.g., PTV or SPH [SFBP09, JFSP10, Thi10]). We validated that our method achieves adequate approximations to time lines and LCS on different input grid types and over long integration intervals. Due to the adaptive material refinement, this scheme reduces the risk of missing features in sparse samplings, compared to established FTLE methods [Hal01a, KPH*09, SP07]. For future work the question arises, whether similar concepts are suited to obtain feature surfaces in 3D time-dependent flow fields. The proposed scheme may open up new possibilities to enhance existing refinement criteria for sampling grids and integral surfaces in flow fields.

Ackknowledgements

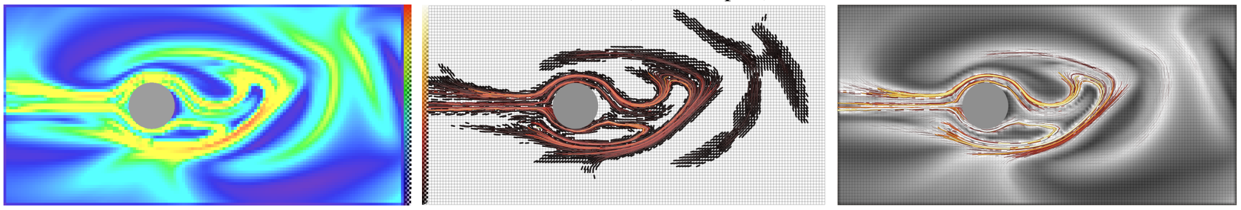
This work was partially funded by the German Federal Ministry of Education and Research under grant number 01LK1213A.



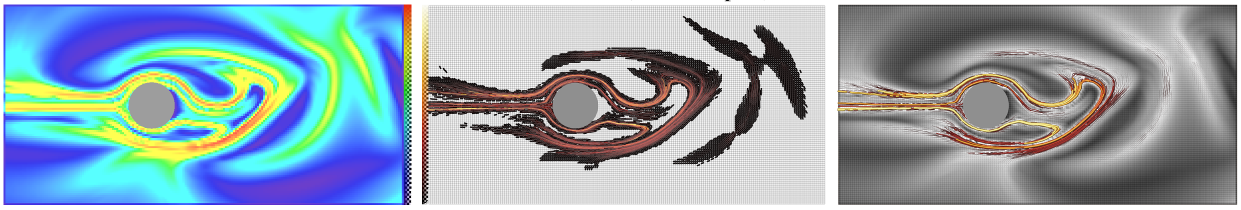
50×25 Grid (1200 samples)



100×50 Grid (5000 samples)

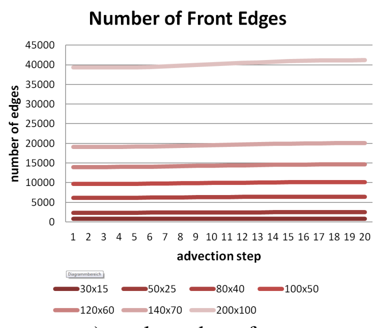


140×70 Grid (9800 samples)

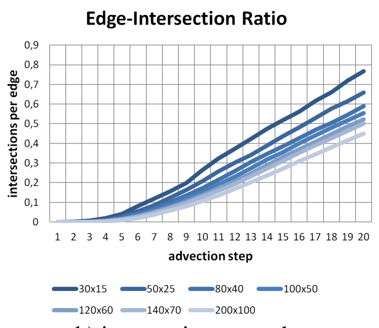


200×100 Grid (20000 samples)

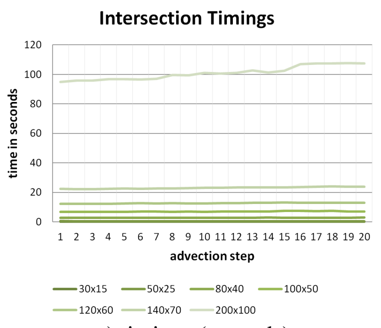
Figure 16: ***Cylinder Flow:*** *Comparison of FTLE [Hal01a] (left column), virtual paths (center, colored by length), and overlay of both (right column, colored and scaled by length) at increasing sampling resolutions for* $t_0 = 0, \tau = 40$.



a) total number of eges



b) intersections per edge



c) timings (seconds)

Figure 17: ***Cylinder Flow:*** *Storage requirements and timings for the resolution series presented in Figure 16: a) shows the total number of edges in the material grid, which grows linearly with ongoing advection; b) the ratio of edge number to number of intersections, which decreases at higher resolutions; c) shows that the timings grow exponentially without further optimization.*

References

[AGJ11] AGRANOVSKY A., GARTH C., JOY K. I.: Extracting Flow Structures Using Sparse Particles. In *Vision, Modeling and Visualization* (2011), Eurographics Association, pp. 153–160. 2, 7

[AS07] ALI S., SHAH M.: A Lagrangian Particle Dynamics Approach for Crowd Flow Segmentation and Stability Analysis. In *Proc. of CVPR* (2007), pp. 1–6. 10

[BJB*11] BHATIA H., JADHAV S., BREMER P.-T., CHEN G., LEVINE J. A., NONATO L. G., PASCUCCI V.: Edge maps: Representing flow with bounded error. In *IEEE Pacific Visualization Symposium (PacificVis)* (2011), pp. 75–82. 6

[BR10] BRUNTON S. L., ROWLEY C. W.: Fast computation of finite-time Lyapunov exponent fields for unsteady flows. *Chaos 20*, 1 (Mar. 2010), 017503. 2

[Ebe96] EBERLY D.: *Ridges in Image and Data Analysis*. Kluwer, 1996. 2, 11

[FH12] FARAZMAND M., HALLER G.: Computing Lagrangian coherent structures from their variational theory. *Chaos 22*, 1 (2012). 2, 3, 4, 8

[GGTH07] GARTH C., GERHARDT F., TRICOCHE X., HANS H.: Efficient computation and visualization of coherent structures in fluid flow applications. *IEEE Trans. on Visualization and Computer Graphics 13* (2007), 1464–1471. 2

[GOPT11] GERMER T., OTTO M., PEIKERT R., THEISEL H.: Lagrangian coherent structures with guaranteed material separation. *Computer Graphics Forum (Proc. EuroVis) 30*, 3 (2011), 761–770. 2, 8

[Hal01a] HALLER G.: Distinguished material surfaces and coherent structures in three-dimensional fluid flows. *Physica D 149* (2001), 248–277. 1, 2, 3, 8, 11, 12

[Hal01b] HALLER G.: Lagrangian structures and the rate of strain in a partition of two-dimensional turbulence. *Physics of Fluids 13*, 11 (2001). 1

[Hal10] HALLER G.: A variational theory of hyperbolic Lagrangian Coherent Structures. *Physica D 240* (2010), 574–598. 1, 2, 8

[HSW11] HLAWATSCH M., SADLO F., WEISKOPF D.: Hierarchical Line Integration. *IEEE Trans. on Visualization and Computer Graphics 17*, 8 (2011), 1148–1163. 2

[JFSP10] JANG Y., FUCHS R., SCHINDLER B., PEIKERT R.: Volumetric evaluation of meshless data from smoothed particle hydrodynamics simulations. In *Proc. Volume Graphics* (2010), pp. 45–52. 10, 11

[KPH*09] KASTEN J., PETZ C., HOTZ I., NOACK B., HEGE H.-C.: Localized finite-time Lyapunov exponent for unsteady flow analysis. In *Vision, Modeling and Visualization* (2009), vol. 1, pp. 265–274. 2, 3, 11

[KRWT12] KUHN A., RÖSSL C., WEINKAUF T., THEISEL H.: A Benchmark for Evaluating FTLE Computations. In *IEEE Pacific Visualization Symposium (PacificVis)* (2012), pp. 121–128. 2, 6

[LM10] LIPINSKI D., MOHSENI K.: A ridge tracking algorithm and error estimate for efficient computation of Lagrangian coherent structures. *Chaos 20*, 1 (2010), 017504. 2

[LR10] LEKIEN F., ROSS S. D.: The computation of finite-time Lyapunov exponents on unstructured meshes and for non-Euclidean manifolds. *Chaos 20*, 1 (2010), 017505. 2, 5

[OPK10] OLCAY A. B., POTTEBAUM T. S., KRUEGER P. S.: Sensitivity of Lagrangian coherent structure identification to flow field resolution and random errors. *Chaos 20*, 1 (2010), 017506. 2

[PD10] PEACOCK T., DABIRI J.: Introduction to focus issue: Lagrangian coherent structures. *Chaos 20*, 1 (2010), 017501. 1, 2

[Pop04] POPINET S.: Free computational fluid dynamics. *ClusterWorld 2*, 6 (2004). 8

[PPF*10] POBITZER A., PEIKERT R., FUCHS R., SCHINDLER B., KUHN A., THEISEL H., MATKOVIC K., HAUSER H.: On the way towards topology-based visualization of unsteady flow - the state of the art. In *EuroGraphics 2010 State of the Art Reports (STARs)* (2010), pp. 137–154. 2

[RGPS05] RUHNAU P., GUETTER C., PUTZE T., SCHNÖRR C.: A variational approach for particle tracking velocimetry. *Measurement Science and Technology 16*, 7 (2005), 1449. 10

[RPH04] RAMSEY S. D., POTTER K., HANSEN C.: Bilinear patch intersections. *J. of Graphics Tools 9(3)* (2004), 41–47. 5, 6, 7

[SEHS11] SENST T., EISELEIN V., HERAS EVANGELIO R., SIKORA T.: Robust modified L2 local optical flow estimation and feature tracking. In *IEEE Workshop on Motion and Video Computing (WMVC)* (2011), pp. 685–690. 3, 10

[SES10] SENST T., EISELEIN V., SIKORA T.: II-LK – a real-time implementation for sparse optical flow. In *Int. Conf. on Image Analysis and Recognition (ICIAR)* (2010), pp. 240–249. 3, 10

[SFBP09] SCHINDLER B., FUCHS R., BIDDISCOMBE J., PEIKERT R.: Predictor-corrector schemes for visualization of smoothed particle hydrodynamics data. *IEEE Trans. on Visualization and Computer Graphics 15*, 6 (October 2009), 1243–1250. 2, 6, 10, 11

[Sha05] SHADDEN S. C.: Lagrangian coherent structures. http://mmae.iit.edu/shadden/LCS-tutorial/, 2005. 3

[SLM05] SHADDEN S., LEKIEN F., MARSDEN J.: Definition and properties of Lagrangian coherent structures from FTLE in two-dimensional aperiodic flows. *Physica D: Nonlinear Phenomena 212*, 3-4 (2005), 271–304. 3, 4, 8

[SP07] SADLO F., PEIKERT R.: Efficient visualization of Lagrangian coherent structures by filtered AMR ridge extraction. *IEEE Trans. on Visualization and Computer Graphics 13*, 6 (2007), 1456–1463. 2, 3, 7, 8, 10, 11

[SP09] SADLO F., PEIKERT R.: Visualizing Lagrangian coherent structures and comparison to vector field topology. *Proc. of TopoInVis* (2009), 15–29. 2

[SPFT11] SCHINDLER B., PEIKERT R., FUCHS R., THEISEL H.: Ridge Concepts for the Visualization of Lagrangian Coherent Structures. *Proc. of TopoInVis* (2011), 1–14. 2, 5, 11

[SRP11] SADLO F., RIGAZZI A., PEIKERT R.: Time-Dependent Visualization of Lagrangian Coherent Structures by Grid Advection. In *Proc. of TopoInVis* (2011), Mathematics and Visualization, Springer, pp. 151–165. 3

[SW10] SADLO F., WEISKOPF D.: Time-Dependent 2-D Vector Field Topology: An Approach Inspired by Lagrangian Coherent Structures. *Computer Graphics Forum 29*, 1 (2010), 88–100. 2, 3

[Thi10] THIFFEAULT J.-L.: Braids of entangled particle trajectories. *Chaos 20*, 1 (2010), 017516. 3, 6, 11

[USE12] ÜFFINGER M., SADLO F., ERTL T.: A time-dependent vector field topology based on streak surfaces. *IEEE Trans. on Visualization and Computer Graphics 19*, 3 (march 2012), 379–392. 2, 3

[USK*12] ÜFFINGER M., SADLO F., KIRBY M., HANSEN C., ERTL T.: FTLE Computation Beyond First-Order Approximation. In *Short Paper Proceedings of Eurographics 2012* (2012), IEEE, pp. 61–64. 2, 3, 10

[WT10] WEINKAUF T., THEISEL H.: Streak lines as tangent curves of a derived vector field. *IEEE Trans. on Visualization and Computer Graphics 16*, 6 (November - December 2010), 1225–1234. VIS. 3

[WT12] WEINKAUF T., THEISEL H.: Flow visualization and analysis using streak and time lines. *Computing in Science and Engineering 14*, 5 (September 2012), 78–84. 3

[WWSS10] WIEBEL A., WANG Q., SCHNEIDER D., SCHEUERMANN G.: Accelerated streak line computation using adaptive refinement. *Journal of WSCG 18* (2010), 17–23. 3